ВИРТУАЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СЕТИ ДЛЯ КОНТЕЙНЕРОВ

VIRTUAL COMPUTING NETWORKS FOR CONTAINERS

M. Kamande
A. Cubahiro

Summary. This article concerns questions related to networking among containers. The case of Linux Containers is analyzed as an example of OS level virtualization. Methods for providing containers with networking in OS Linux are discussed. The networking questions are discussed for the case of virtual computational cluster. Two approaches for networking are also discussed. In particular, these virtual interfaces provided in OS Linux are discussed: «veth» interface and «macvlan» interface.

Keywords: Containers, networking, virtual networks, computational clusters, Linux Containers.

Каманде Магдалине Вамбуи
Аспирант, Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)
magdalynde@gmail.com
Чубахиро Амисси
Аспирант, Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)
amcubahiro@gmail.com

Аннотация. Рассматриваются вопросы сетевого взаимодействия в рамках контейнеров. Проанализирован случай виртуализации на уровне операционной системы с использованием Linux Containers. Рассмотрены способы организации сетевого взаимодействия для контейнеров в операционной системе Linux. Вопросы сетевого взаимодействия рассматриваются применительно для случая виртуального вычислительного кластера. Рассмотрена работа двух виртуальных интерфейсов для ОС Linux: «veth» и «macvlan».

Ключевые слова: Контейнеры, сетевое взаимодействие, виртуальные сети, вычислительные кластеры, Linux Containers.

Общие сведения

Виртуализация на уровне ОС — перспективное направление, востребованное в последние годы. Такой подход к виртуализации вызывает все больший интерес как в академической среде, так в сфере бизнеса. Контейнеры в ОС Linux — одно из лучших решений. Доказательством этому могут служить программные комплексы, созданные на их основе (например, Docker [1], Singulartity [2]). Виртуализация на уровне ОС предлагает достаточную степень изоляции ресурсов при крайне низких накладных расходах. Такое соотношение не может предложить ни один другой тип виртуализации. Контейнеры в ОС Linux предлагают гибкость конфигурирования и производительность для решения различных задач. Контейнеры могут использоваться как для запуска сервисов, так и для организации расчетов. И в том, и в другом случае производительность сети — важный фактор, который может сказаться на общей производительности расчетов, ведь как для предоставления сервиса, так и для расчетов, очень часто используют вычислительные кластеры, в рамках которых производительность сети выходит на первый план, зачастую являясь главным ограничивающим фактором. Таким образом, рассмотрение вопросов, связанных с виртуальными вычислительными сетями для контейнеров, является важной и актуальной задачей. Рассмотрению вопросов производительности и безопасности контейнеров и виртуальных машин посвящен ряд статей, например, [3] и [4].

Но, прежде чем рассматривать отдельные аспекты работы виртуальных сетей, необходимо рассмотреть возможности изоляции ресурсов в рамках контейнеров Linux, они представлены отдельными пространствами имен, которые перечислены далее.

1. Отдельное пространство имен для процессов — PID («Process Identifier») namespace. Использование данного пространства имен позволяет изолировать иерархию процессов в рамках отдельных контейнеров. Так, процессы одного контейнера будут недоступны для процессов другого контейнера (пользователь, работающий в рамках одного контейнера, даже не сможет увидеть процессы другого контейнера).
2. Отдельное пространство имен для межпроцессного взаимодействия — IPC («Inter-Process Communication») namespace. Позволяет изолировать ресурсы IPC.
3. Отдельное пространство имен для пользователей и групп — User namespace. Позволяет изолировать ресурсы, связанные с обеспечением безопасности работы пользователей и групп.
4. Отдельное пространство имен для иерархии каталогов — Mount namespace. Позволяет разным контейнерам работать с различными иерархиями каталогов.
5. Отдельное пространство имен для UTS («UNIX Timesharing System») — UTS namespace. Позволяет изолировать ресурсы, связанные с именем узла.
6. Отдельное пространство имен для сетевого взаимодействия (сетевое пространство имен) — Net

namespace. Позволяет изолировать ресурсы, связанные с обеспечением работы сети.

Выбор необходимой степени изоляции выполняется при создании нового процесса с помощью системного вызова «clone». В рамках данной статьи рассматривается лишь вариант использования отдельных пространств имен для сетевого взаимодействия. Большая степень изоляции обеспечиваться не будет, поскольку данная статья посвящена именно виртуальным сетям для контейнеров.

Отдельное сетевое пространство имен в рамках Linux позволяет обеспечить изоляцию следующих ресурсов: настройки отдельных стеков сетевых протоколов (например, ограничения на использование памяти для очередей, связанных с сокетами), ресурсы, связанные со стеками протоколов (например, таблицы маршрутизации, адреса и номера портов), сетевые интерфейсы. Стоит отметить, что сетевые интерфейсы могут быть как физическими, так и виртуальными — и в том, и в другом случае в рамках Linux сетевой интерфейс будет представлен экземпляром структуры «net_device». И каждое такое представление связано с определенным сетевым пространством имен (созданным или пространством имен по умолчанию). При этом сетевое устройство может принадлежать лишь одному пространству имен. Так, сетевые пространства имен определяют к каким сетевым устройствам будет иметь доступ контейнер.

Способы организации сетевого взаимодействия для контейнеров

Однако выделение физического интерфейса для контейнера в отдельном сетевом пространстве имен кажется нецелесообразным, ведь в таком случае доступ к нему будет иметь лишь один контейнер. Поэтому для контейнеров, как правило, используются виртуальные сетевые интерфейсы. Отдельные сетевые интерфейсы для каждого контейнера позволяют работать контейнерам с отдельными сетевыми адресами (возможно, в разных подсетях). Использование разных сетевых пространств имен для разных контейнеров позволяет задавать настройки для сетевого стека каждого контейнера независимо. В рамках Linux доступен ряд виртуальных сетевых интерфейсов, однако для работы контейнеров, как правило, используют или интерфейсы «veth» («Virtual Ethernet») вместе с интерфейсом «bridge», или интерфейсы «macvlan». Именно рассмотрению этих двух вариантов и будет уделено основное внимание в рамках данной статьи. Отсюда и далее под «контейнером» подразумевается контейнер, работающий в отдельном сетевом пространстве имен, если явно не указано иное.

Интерфейс «veth»

Данный интерфейс подразумевает пару интерфейсов одинакового типа. При отправке кадра через один из двух интерфейсов, кадр появляется на втором интерфейсе — ОС получает кадр примерно также, как если бы он был получен на втором интерфейсе, например, из внешней сети. Интерфейсы равноценны, для их работы используются одни и те же функции. Если интерфейсы принадлежат к разным сетевым пространствам имен, то с помощью пары таких интерфейсов можно организовать передачу кадров между отдельными сетевыми пространствами имен. Такой подход используется как раз в рамках контейнеров Linux с разными сетевыми пространствами имен. Благодаря интерфейсу «veth» можно организовать передачу данных между отдельными контейнерами или между контейнером и хостовой ОС (имеется ввиду сетевое пространство имен по умолчанию, в котором работают процессы, не принадлежащие контейнерам).

Однако отдельная пара интерфейсов «veth» сама по себе, как правило, не используется. Контейнерам, как правило, требуется доступ во внешнюю сеть. Для этого необходимо организовать передачу данных между контейнером и физическим сетевым интерфейсом. Для этой цели, как правило, используется виртуальный мост Linux — интерфейс «bridge». Он выполняет ту же роль, что и обыкновенный физический мост, а именно объединение отдельных сегментов сети. Он точно также работает на втором уровне модели OSI («Open System Interconnection»). Он постепенно собирает информацию об известных узлах с тем, чтобы отправлять кадры лишь по нужному порту. Он поддерживает STP («Spanning Tree Protocol»). Как правило, мост работает в сетевом пространстве имен по умолчанию, в котором, как правило, работают и физические сетевые интерфейсы. Поэтому к мосту можно подключить, как физический интерфейс, так и интерфейс «veth» из пары, работающий в сетевом пространстве имен по умолчанию (речь идет об ассоциации экземпляра структуры «net_device», представляющего физический сетевой интерфейс в системе, с виртуальным сетевым мостом). Если второй интерфейс будет работать в сетевом пространстве имен контейнера, то благодаря такому подключению можно будет наладить передачу данных из контейнера во внешнюю сеть и наоборот. Схематично работа интерфейсов «veth» и «bridge» изображена на рис. 1.

На рис. 1 «veth0» и «veth1» являются парными интерфейсами «veth», как и «veth2» и «veth3». Интерфейсы «veth1» и «veth3» находятся в отдельных сетевых пространствах имен, используемых в рамках контейнера 1 и 2 соответственно. Интерфейсы «veth0» и «veth2» находятся в сетевом пространстве имен по умолчанию

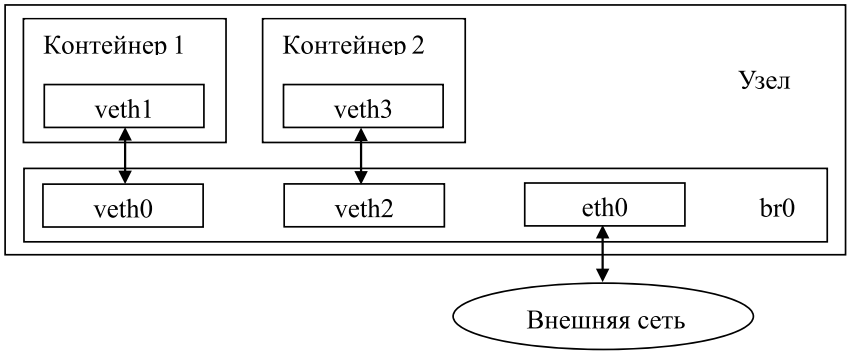

Рис. 1

и включены в мост «br0» (интерфейс «bridge»). В этот же мост включен и физический сетевой интерфейс «eth0», подключенный ко внешней сети.

При получении кадра из внешней сети через интерфейс, включенный в мост, ОС отдает кадр на обработку специальной функции реализации моста. В рамках этой функции определяется порт назначения для кадра по MAC-адресу назначения в заголовке Ethernet. Если это MAC-адрес интерфейса «veth» контейнера, мост передаст кадр на порт, с которым связан второй интерфейс «veth», и кадр в итоге появится на интерфейсе в рамках контейнера. Таким образом контейнер получит кадр из внешней сети.

Интерфейс «macvlan»

Данный интерфейс, в отличие от интерфейса «veth», не требует дополнительного интерфейса-моста, так как сам может выполнять функции моста. При работе с данным интерфейсом необходимо указать «нижележащее» устройство. Таким устройством может быть как физический и интерфейс, так и виртуальный. С одним «нижележащим» устройством может быть связано несколько интерфейсов «macvlan».

Интерфейс «macvlan» может работать в нескольких режимах, которые перечислены далее.

1. Режим «private». В данном режиме каждый интерфейс «macvlan» работает независимо от других интерфейсов того же типа. Такой режим подходит, например, если предполагается работа лишь одного контейнера.

2. Режим «passthru». В данном режиме с одним «нижележащим» устройством может быть связан только один интерфейс «macvlan». В этом режиме все кадры, пришедшие на «нижележащее» устройство, будут передаваться на этот единственный интерфейс «macvlan» (как если бы они были физически получены на данном интерфейсе «macvlan»). В этом режиме виртуальный интерфейс «macvlan» даже унаследует MAC-адрес «нижележащего» устройства.

3. Режим «bridge». В данном режиме все виртуальные интерфейсы «macvlan», связанные с одним «нижележащим» устройством, могут свободно пересылать кадры друг другу, при этом такие кадры будут передаваться без участия «нижележащего» устройства, в рамках одного узла. «Нижележащее» устройство будет задействовано, когда потребуется отправить кадр от одного из виртуальных интерфейсов «macvlan» во внешнюю сеть.

4. Режим «VEPA» («Virtual Ethernet Port Aggregator»). Данный режим выбирается по умолчанию. Это специальный режим, в рамках которого передача данных осуществляется через «нижележащее» устройство, даже если кадр передается от одного интерфейса «macvlan» другому. При этом предполагается, что кадр, адресованный другому интерфейсу «macvlan», а не узлу во внешней сети, будет передан обратно на узел внешним оборудованием. Такой режим работы может понадобится в случае, когда требуется централизованная обработка всех кадров, даже если они передаются от одного контейнера другому. Например, по причинам безопасности. В этом случае, вместо применения дополнительных настроек безопасности на узле с контейнерами, можно передавать кадры через общий коммутатор, на котором уже выполнены все необходимые настройки. Но при этом потребуется специальный коммутатор, который будет возвращать кадры на узел, если они адресованы виртуальному интерфейсу в рамках этого же узла (обычный коммутатор, как раз наоборот, не должен отправлять кадр на порт, с которого он пришел).

5. Режим «source». Специальный режим, в котором выполняется работа с отдельным список MAC-адресов (для поиска используется MAC-адрес источника из кадра).

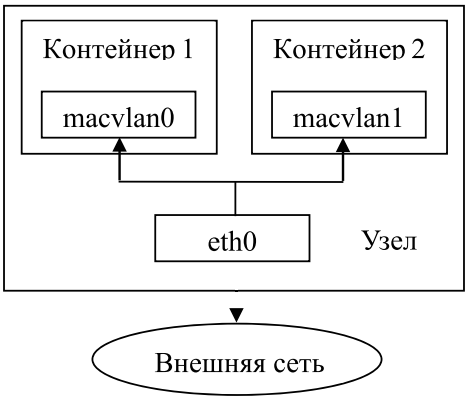

Рис. 2

В случае работы нескольких контейнеров без специального коммутатора, поддерживающего «VEPA», во внешней сети, можно использовать режим «bridge». Он и будет рассмотрен далее. Для работы каждого из контейнеров создается отдельный виртуальный интерфейс «macvlan», расположенный в отдельном сетевом пространстве имен (связанным с контейнером), однако, все эти виртуальные интерфейсы связаны с одним «нижележащим» устройством, которое, как правило, является физическим интерфейсом. При такой организации работы кадры могут свободно передаваться между отдельными контейнерами в рамках одного узла, а кадры для узлов во внешней сети могут быть переданы через «нижележащее» устройство, как и кадры от таких узлов в обратном направлении. Схематично работа интерфейса «macvlan» изображена на рис. 2.

На рис. 2 «macvlan0» и «macvlan1» являются интерфейсами «macvlan», связанными с общим «нижележащим» устройством — физическим сетевым интерфейсом «eth0». Интерфейсы «macvlan0» и «macvlan1» находятся в отдельных сетевых пространствах имен, используемых в рамках контейнера 1 и 2 соответственно.

При получении кадра на «нижележащем» устройстве выполняется поиск устройства-назначения (виртуального устройства «macvlan») по MAC-адресу назначения заголовка Ethernet поступившего кадра. Если такое устройство найдено, кадр передается на это устройство (в режиме «passthru» кадр просто передается на единственное устройство «macvlan», связанное с данным «нижележащим» устройством). Таким образом контейнер получает кадр из внешней сети. В случае отправки кадра с устройства «macvlan» в режиме «bridge», выполняется поиск устройства-назначения по MAC-адресу назначения заголовка Ethernet отправляемого кадра. Если такое устройство найдено, и оно также работает в режиме «bridge», выполняется передача кадра на данное устройство, в другом случае кадр передается во внешнюю сеть через «нижележащее» устройство. Таким образом контейнер может передать кадр во внешнюю сеть. В случае multicast Ethernet кадров устройство «macvlan» в режиме «bridge» передает кадры всем устройствам «macvlan», работающим в этом же режиме и связанным с этим же «нижележащим» устройством, а также во внешнюю сеть через «нижележащее» устройство.

Стоит также отметить, что если потребуется сетевое взаимодействие (например, доступ по ssh) контейнера (с интерфейсом «macvlan» в режиме «bridge») с хостовой ОС (сетевым пространством имен по умолчанию), будет необходимо создать интерфейс «macvlan» (работающий в режиме «bridge») в сетевом пространстве имен по умолчанию, связанный с тем же «нижележащим устройством», иначе доступ к контейнеру по сети будет возможен лишь из внешней сети (хотя запуск приложений в контейнере с отдельным сетевым пространство имен все также будет возможен, например, через «ip netns exec»).

Тестирование виртуальных интерфейсов для контейнеров

Для данной статьи был выполнен ряд тестов, как синтетических, так и предполагающих запуск прикладного приложения для расчетов. Требовалось оценить накладные расходы, связанные с сетью, для случая использования контейнеров в рамках вычислительных кластеров. Данное направление вызывает все больший интерес в последнее время (например, в [4] рассматривается как раз такой вариант). Именно поэтому основное внимание было уделено прикладным задачам.

Во всех случаях выполнялся запуск тестов на двух отдельных узлах (в контейнерах или без них). В рамках контейнеров использовались лишь отдельные сетевые пространства имен. Для обеспечения контейнеров сетью на каждом из узлов использовался сначала интерфейс

Таблица 1

	iperf, Мбит/сек	LINPACK, % от NIC	GROMACS, сек
NIC	94.1	100	242.364
Veth	94.016	99.1	242.452
Macvlan	94.033	99.7	242.944

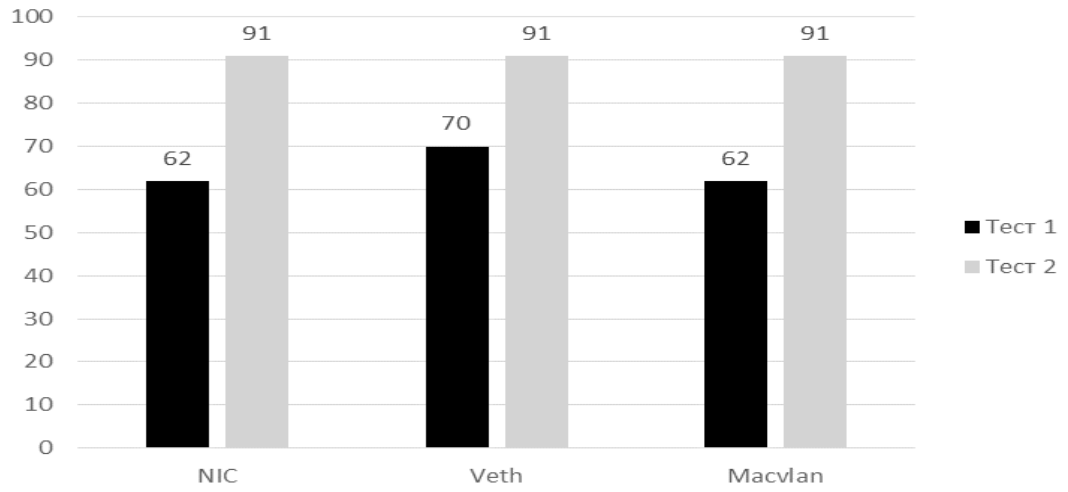

Рис. 3

«veth» вместе с интерфейсом «bridge», а затем интерфейс «macvlan». Затем выполнялось сравнение полученных результатов при использовании каждого из вариантов.

В рамках первого варианта интерфейс «bridge» находился в сетевом пространстве имен по умолчанию. В интерфейс «bridge» включалось одно из устройств «veth», другое устройство «veth» использовалось в рамках контейнера (находилось в сетевом пространстве имен, связанном с контейнером). Также в интерфейс «bridge» включался физический сетевой интерфейс.

В рамках второго вариант использовался интерфейс «macvlan» в режиме «bridge» в сетевом пространстве имен контейнера. «Нижележащим» устройством являлся физический сетевой интерфейс.

Для оценки производительности выполнялся запуск тестов с теми же параметрами не только в рамках контейнера, но и на хостовой ОС, без использования контейнеров (в сетевом пространстве имен по умолчанию) на физическом сетевом интерфейсе.

Создание сетевого пространства имен выполнялось при помощи программы «ip» («ip netns add»). Для контейнеров использовалась отдельная подсеть IP. Настройки параметров сетевого стека выбирались по умолчанию для данной ОС (описание доступных параметров стека TCP/IP приводится в [5]). И на узлах, и в рамках контейнеров использовалась система «Fedora 26». Использовались узлы архитектуры x86_64 с 8-ядерным процессором и 4 ГБ оперативной памяти. Для соединения узлов использовался коммутатор Ethernet, поддерживающий скорость передачи данных до 100 Мбит/сек (Fast Ethernet).

Сначала выполнялся стандартный тест для оценки производительности сети — тест «iperf». В табл. 1 приведены результаты запуска теста (скорость передачи данных в Мбит/сек) с параметрами по умолчанию для случая физического интерфейса (отсюда и далее на рисунках для него будет дано обозначение «NIC»), интерфейса «veth» и интерфейса «macvlan».

Далее выполнялся стандартный тест LINPACK на 2 узлах, запускалось по 8 процессов на узел (по одному процессу на ядро). Результаты его выполнения также приведены в табл. 1. Производительность для случая использования физического сетевого интерфейса без виртуальных сетевых интерфейсов была взята за основу (100%). Для различных виртуальных интерфейсов приводится процент от производительности, полученной для варианта использования только физического сетевого интерфейса.

Наконец, в четвертом столбце табл. 1 приведено время выполнения тестовой задачи GROMACS [6]. Запускалось по 1 процессу на узле, в рамках каждого процесса запускалось по 8 потоков: в рамках GROMACS использовался вариант MPI («Message Passing Interface») вместе с OpenMP («Open Multi-Processing»).

Как видно, во всех трех случаях потери производительности, по сути, нет. Скорость передачи данных, измеренная при помощи программы «iperf», между контейнерами с использованием интерфейса «veth» практически не отличается от скорости в случае использования «macvlan» и от случая передачи данных между узлами без использования контейнеров и виртуальных интерфейсов. Отличия минимальны — их можно объяснить влиянием случайных, несущественных факторов.

В случае выполнения теста LINPACK и тестовой задачи GROMACS потери производительности расчетов не наблюдалось, различия в производительности также были незначительны и также могли быть объяснены влиянием случайных факторов. Наконец, на рис. 3 приводятся результаты запуска (время выполнения в секундах) тестовых задач OpenFOAM [7] для солверов «interFoam» («тест 1») и «rhoPimpleFoam» («тест 2»).

В случае тестовой задачи для солвера «rhoPimpleFoam» потери производительности не наблюдалось. Однако небольшая потеря производительности присутствует в случае тестовой задачи для солвера «interFoam» при использовании интерфейса «veth». После детального анализа выяснилось, что эта разница во времени может быть связана с количеством пакетов: в случае использования интерфейса «macvlan» стеку TCP/IP удается чаще объединять данные для пересылки и отправлять в рамках пакета большей длины. Как результат — в сеть выдается меньше пакетов. Передаваемые пакеты имеют большую длину, благодаря этому снижаются накладные расходы. Это различие между «macvlan» и «veth» может быть объяснено внутренними проверками стека и особенностями работы «veth»: функция-деструктор вызывается при обработке в рамках «veth».

В статье [3] приводится сравнение производительности для случая сети Ethernet 10 Gbit. В упомянутой статье снижения производительности при использовании Docker с виртуальным интерфейсом не наблюдалось. Как уже было сказано, наблюдаемое на одном из тестов в рамках данной статьи различие в производительности может быть объяснено внутренними особенностями работы стека и не должно проявляться при отправке совершенно одинакового количества пакетов в сеть. Однако требуется больше тестов, соответствующих различным шаблонам использования сети, чтобы сделать окончательный вывод.

Выводы

Таким образом, в рамках данной статьи после выполнения описанных тестов при использовании виртуальных сетевых интерфейсов для контейнеров снижения производительности сетевого взаимодействия практически не наблюдалось. Интерфейсы «veth» и «macvlan» практически не отличаются по производительности сети (за исключением особых случаев, разница в которых может быть объяснена внутренними особенностями работы интерфейсов и сетевого стека). В случае использования «veth», как правило, требуется создание интерфейса «bridge». В случае использования «macvlan» создание дополнительного интерфейса не требуется. Также «macvlan» поддерживает несколько режимов работы, в том числе «VEPA». Все это делает интерфейс «macvlan» более привлекательным для типичного варианта виртуальной сети контейнеров.

ЛИТЕРАТУРА

1. Официальный сайт проекта Docker [Электронный ресурс] // https://www.docker.com/
2. Официальный сайт проекта Singularity [Электронный ресурс] // http://singularity.lbl.gov/
3. Felter W. et al. An Updated Performance Comparison of Virtual Machines and Linux Containers [Текст] / W. Felter, A. Ferreira, R. Rajamony, J. Rubio // Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On.— IEEE, 2015.— p. 171–172.
4. Xavier M. G. et al. Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments [Текст] / M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, C. A. F. De Rose // Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on.— IEEE, 2013.— p. 233–240.
5. Параметры стека TCP/IP в ОС Linux [Электронный ресурс] // https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt
6. Hess B. et al. GROMACS4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation [Текст] / B. Hess, C. Kutzner, D. van der Spoel, E. Lindahl // Journal of chemical theory and computation 4.3, 2008.— p. 435–447.
7. Официальный сайт проекта OpenFOAM [Электронный ресурс] // https://www.openfoam.com/

Журнал «Современная наука: актуальные проблемы теории и практики»